Appendix A: statistical regression models

Linear and log-linear regression

Most commonly, ordinary least squares multivariate regression of the follow-up FDG PET images can be used to fit coefficients that are linear in the pre-treatment PET covariates:

$$\textit{Linear} \qquad FDG_{\text{post}} = \beta_0 + \beta_{\text{FDG}} FDG_{\text{pre}} + \beta_{\text{FLT}} FLT_{\text{pre}} + \beta_{\text{CuATSM}} CuATSM_{\text{pre}} + error \qquad \text{(A1)}$$

The relationship is assumed to hold for each tumour voxel of the registered PET images; this could be indicated by an additional subscript, which is dropped here for clarity. The set of coefficients may be estimated by a least squares fit of the model to the observed FDG uptake values in the post scan and evaluated for goodness-of-fit by the coefficient of determination R^2.

This simple model requires linearity and approximately constant variance on the chosen scale. Patients with a complete or near-complete regional PET response will have a follow-up FDG uptake distribution at three months that is skewed towards low values with few high uptake regions. In order to place more weight on higher image voxel values and remove this skew, a logarithmic transformation was applied, yielding a log-linear model with multiplicative error:

$$\textit{Log-linear} \qquad FDG_{\text{post}} = 10^{\beta_0} \cdot FDG_{\text{pre}}{}^{\beta_{\text{FDG}}} \cdot FLT_{\text{pre}}{}^{\beta_{\text{FLT}}} \cdot CuATSM_{\text{pre}}{}^{\beta_{\text{CuATSM}}} \cdot 10^{error} \qquad \text{(A2)}$$

The log-linear model fits the relative rather than the absolute change in the PET signal. This model incorporates error on the log scale, which means that both signal and noise grow at the same rate. However, many sources of uncertainty in PET images that are physical (e.g. photon counting statistics), biological (e.g. dynamics of tracer delivery), and technical (e.g. registration of images) in nature may not contribute error on the same linear scale as the signal. Without a means of precisely characterizing the mathematical form of these uncertainty sources, a simple link function between signal and error can be devised through generalized linear regression. This yields a generalized linear model (GLM) with a log-linear link function between signal and error:

$$\begin{matrix}\textit{Generalized}\\ \textit{linear}\end{matrix} \qquad FDG_{\text{post}} = 10^{\beta_0} \cdot FDG_{\text{pre}}{}^{\beta_{\text{FDG}}} \cdot FLT_{\text{pre}}{}^{\beta_{\text{FLT}}} \cdot CuATSM_{\text{pre}}{}^{\beta_{\text{CuATSM}}} + error \qquad \text{(A3)}$$

These models all characterize the regional imaging response relationship between pre-treatment PET images and post-treatment FDG PET images as a fit to a single function. They may also incorporate additional features, such as interaction and other higher order terms; however, these turned out to be insignificantly different from zero in the present study (data not shown).

Linear mixed-fit regression

The previous regression models all assume that a single mathematical relationship links pre-treatment PET image uptake and post-treatment FDG PET uptake of all tumour voxels in a given patient. However, our data analyses suggested that this was not a reasonable assumption in some tumours. Instead, a mixed regional PET imaging response in individual patients due to intra-tumour variation can be fit by multiple functions. A simple formalism for such a model is a set of two linear regressions describing the "complete responding" (CR) and "non-responding" (NR) voxel subpopulations. These regional PET imaging response subpopulations are defined post-hoc since the groups of voxels are not classified *a priori*. In the linear mixture model, the voxels are weighted to the fit of each line in proportion to the inverse squared errors between the observed value and the predicted value on each line. For simplicity, we consider only the pre-treatment FDG covariate in the regression and without loss of generality require that the CR line have a smaller slope than the NR line:

$$\textit{Linear Mixture Model}\quad \begin{aligned} FDG_{\text{post, CR}} &= \beta_{0,\text{CR}} + \beta_{\text{FDG,CR}} \cdot FDG_{\text{pre}} + error \\ FDG_{\text{post, NR}} &= \beta_{0,\text{NR}} + \beta_{\text{FDG,NR}} \cdot FDG_{\text{pre}} + error \end{aligned}, \text{ where } \beta_{\text{FDG, CR}} < \beta_{\text{FDG, NR}} \tag{A4}$$

The coefficients are estimated by weighted least squares, whereby the residuals of each fit form the basis for the weights of proportionality:

$$\begin{aligned} w_{\text{CR}} &= \frac{\left(FDG_{\text{post}} - FDG_{\text{post, NR}}\right)^2}{\left(FDG_{\text{post}} - FDG_{\text{post, NR}}\right)^2 + \left(FDG_{\text{post}} - FDG_{\text{post, CR}}\right)^2} \\ w_{\text{NR}} &= 1 - w_{\text{CR}} \end{aligned} \tag{A5}$$

Since the weights are functions of the fitted FDG_{post} values, and the latter are computed using the weights, the estimation process is iterative. Convergence of the computations occurs when all estimated coefficients differ from their prior estimate by 10^{-6} or less. A weighted R^2, which measures the total percentage of the observed variance σ^2 in FDG_{post} that can be explained by both fits of the linear mixture model, is defined as the goodness-of-fit metric:

$$R^2_{\text{total}} = w_{\text{CR}} \cdot R^2_{\text{CR}} + w_{\text{NR}} \cdot R^2_{\text{NR}} = \frac{w_{\text{CR}} \cdot \sigma^2\left(FDG_{\text{post, CR}}\right) + w_{\text{NR}} \cdot \sigma^2\left(FDG_{\text{post, NR}}\right)}{\sigma^2\left(FDG_{\text{post}}\right)} \tag{A6}$$

The presumed responding and non-responding voxels within each tumour are modelled separately. This separation is stochastic in that the two subpopulations are estimated only by the relative weights w_{CR} and w_{NR}. For example, a voxel could be estimated as having a 76 percent chance of being in the non-responding group and a 24 percent chance of being in the responding group.

Appendix B: canine sinonasal cancer patient PET/CT images and scatterplots

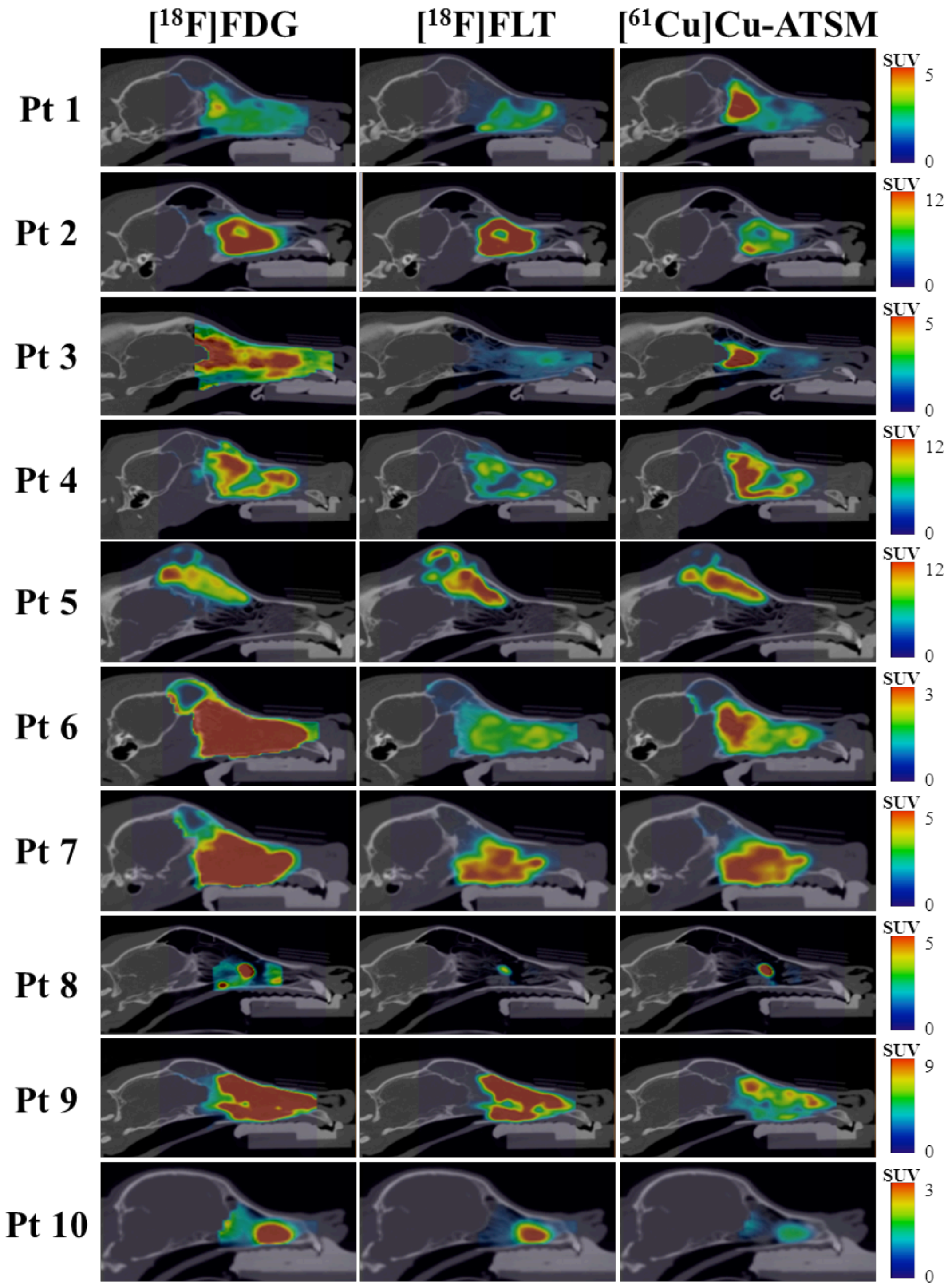

Figure B1. *Fused PET/CT images of pre-treatment FDG, FLT and Cu-ATSM uptake in ten canine sinonasal cancer patients.* Uptake distributions are shown in the planning target volume and the colourbars are fixed for each patient.

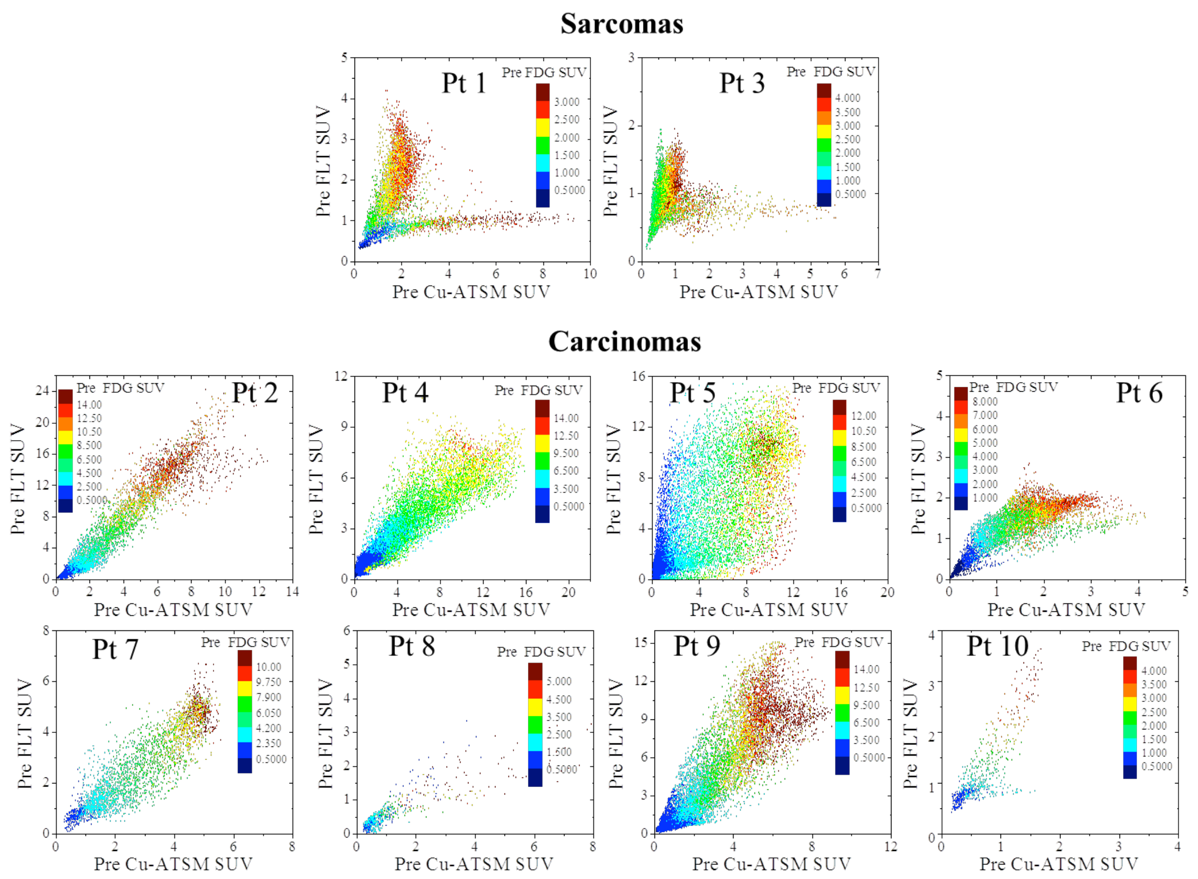

Figure B2. *PET image voxel colourwashed scatter plots of pre-treatment FDG, FLT and Cu-ATSM uptake in ten canine sinonasal cancer patients.* The patients have been grouped by histology to highlight differences in absolute standardized uptake values (SUV) as well as differences in the number of modes or branches in linear correlation between the three radiotracers.

Appendix C: multivariate voxel regression R code

Linear models, log-linear models with multiplicative errors, and generalized linear models with additive error via a log link function

```
# Linear models, log-linear models with multiplicative errors, GLMs with
# additive errors, log link

ii <- c(1,2,3,5,6,8,9,10,11)
resp <- rep("Nonresponse", length(ii))
resp[(ii==2) | (ii==6) | (ii==8) | (ii==9)] <- "Response"
jj <- rep("NR", length(ii))
jj[(ii==2) | (ii==6) | (ii==8) | (ii==9) | (ii==10) | (ii==11)] <- "R "
jj <- paste(ii, jj)

# Generalized linear model with log link, additive errors
# "LLAE" = log-linearity, additive errors with constant variance
# with FDG0, FLT0, and CuATSM0

coefs.LLAE <- NULL
rsquared.LLAE <- NULL
for (i in ii) {
print(i)
FDG3     <- as.numeric(evil(i)$FDG3)

log10FDG0     <- log10(as.numeric(evil(i)$FDG0))
Slog10FDG0    <- log10FDG0 - mean(log10FDG0)

# cludge to adjust for negative FLT0s:

uuFLT0     <- as.numeric(evil(i)$FLT0)
uuFLT0[uuFLT0 <= 0]     <- .0365
log10FLT0     <- log10(uuFLT0)
# log10FLT0     <- log10(as.numeric(evil(i)$FLT0))
Slog10FLT0    <- log10FLT0 - mean(log10FLT0)

log10CuATSM0  <- log10(as.numeric(evil(i)$CuATSM0))
Slog10CuATSM0 <- log10CuATSM0 - mean(log10CuATSM0)

rr <- glm(FDG3 ~ Slog10FDG0 + Slog10CuATSM0 + Slog10FLT0,
        family=gaussian(link = "log"))

coefs.LLAE <- rbind(coefs.LLAE, rr$coefficients)
rsquared.LLAE <- c(rsquared.LLAE,
                (rr$null.deviance-rr$deviance)/rr$null.deviance)
            }

p.LLAE <- NULL
for (i in 1:dim(coefs.LLAE)[2]) {
```

```
p.LLAE <- c(p.LLAE, t.test(coefs.LLAE[,i])$p.value)
                        }

# Linear model with log link, multiplicative errors
# "LLME" = log-linearity, multiplicative errors with constant variance
# with FDG0, FLT0, and CuATSM0

coefs.LLME <- NULL
rsquared.LLME <- NULL
for (i in ii) {
log10FDG3     <- log10(as.numeric(evil(i)$FDG3))

log10FDG0     <- log10(as.numeric(evil(i)$FDG0))
Slog10FDG0    <- log10FDG0 - mean(log10FDG0)

uuFLT0     <- as.numeric(evil(i)$FLT0)
uuFLT0[uuFLT0 <= 0]     <- .0365
log10FLT0     <- log10(uuFLT0)
# log10FLT0     <- log10(as.numeric(evil(i)$FLT0))
Slog10FLT0    <- log10FLT0 - mean(log10FLT0)

log10CuATSM0  <- log10(as.numeric(evil(i)$CuATSM0))
Slog10CuATSM0 <- log10CuATSM0 - mean(log10CuATSM0)

rr <- lm(log10FDG3 ~ Slog10FDG0 + Slog10CuATSM0 + Slog10FLT0)

coefs.LLME <- rbind(coefs.LLME, rr$coefficients)
rsquared.LLME <- c(rsquared.LLME, summary(rr)$r.squared)
            }

p.LLME <- NULL
for (i in 1:dim(coefs.LLME)[2]) {
p.LLME <- c(p.LLME, t.test(coefs.LLME[,i])$p.value)
                        }

# Linear model with identity link, additive errors
# "LAE " = log-linearity, multiplicative errors with constant variance
# with FDG0, FLT0, and CuATSM0

coefs.LAE  <- NULL
rsquared.LAE  <- NULL
for (i in ii) {
FDG3     <- as.numeric(evil(i)$FDG3)

FDG0     <- as.numeric(evil(i)$FDG0)
SFDG0    <- FDG0 - mean(FDG0)

FLT0     <- as.numeric(evil(i)$FLT0)
SFLT0    <- FLT0 - mean(FLT0)
```

```
CuATSM0  <- as.numeric(evil(i)$CuATSM0)
SCuATSM0 <- CuATSM0 - mean(CuATSM0)

rr <- lm(FDG3 ~ SFDG0 + SCuATSM0 + SFLT0)

coefs.LAE  <- rbind(coefs.LAE , rr$coefficients)
rsquared.LAE  <- c(rsquared.LAE , summary(rr)$r.squared)
            }

# Print out results
p.LAE  <- NULL
for (i in 1:dim(coefs.LAE )[2]) {
p.LAE  <- c(p.LAE , t.test(coefs.LAE [,i])$p.value)
                        }

cbind(c(jj, NA), round(rbind(coefs.LAE, p.LAE), 3),
c(round(rsquared.LAE,2), NA))

# Print out results
p.LLME  <- NULL
for (i in 1:dim(coefs.LLME )[2]) {
p.LLME  <- c(p.LLME , t.test(coefs.LLME [,i])$p.value)
                        }

cbind(c(jj, NA), round(rbind(coefs.LLME, p.LLME), 3),
c(round(rsquared.LLME,2), NA))

# Print out results
p.LLAE  <- NULL
for (i in 1:dim(coefs.LLAE )[2]) {
p.LLAE  <- c(p.LLAE , t.test(coefs.LLAE [,i])$p.value)
                        }

cbind(c(jj, NA), round(rbind(coefs.LLAE, p.LLAE), 3),
c(round(rsquared.LLAE,2), NA))
```

Linear mixture models

```
# Algorithm for fitting mixture lms (upgradeable to glm, but currently
# written for lm only) with two components

ii <- c(1,2,3,5,6,8,9,10,11)
resp <- rep("Nonresponse", length(ii))
resp[(ii==2) | (ii==6) | (ii==8) | (ii==9)] <- "Response"
jj <- rep("NR", length(ii))
jj[(ii==2) | (ii==6) | (ii==8) | (ii==9) | (ii==10) | (ii==11)] <- "R "
jj <- paste(ii, jj)

# Plot FDG3 vs. FDG0 with smoothers
```

```
postscript("ps.out1", horizontal=T)
par(mfrow=c(2,1), oma=c(1.5,2,2,1), mar=c(4,4,4,0.5))
for (i in (1:length(ii))) {
xx <- as.numeric(evil(ii[i])$FDG0)
yy <- as.numeric(evil(ii[i])$FDG3)
uu <- range(xx, yy)
plot(xx, yy, xlim=uu, ylim=uu, pch=".",
    main=paste("CDP00", jj[i], sep=""), xlab="FDG0", ylab="FDG3")
abline(c(0,1), lty=2)
mtext("FDG3 vs. FDG0", outer=T, cex=2, col="red")
lines(lowess(xx, yy, f=.1,), col="blue")
# readline("next?")
                    }

mixreg <- function(yy, xx, multiplier=1, tolerance=.000001) {
# I. UNIVARIATE x=FDG0; y=FDG3; parameters alpha1, alpha2, beta1, beta.

# Generate starting values for alpha1, alpha2, beta1, beta.
reg0 <- lm(yy ~ xx)
alpha0 <- reg0$coefficients[1]
beta0  <- reg0$coefficients[2]
# Generate two pairs of (alpha, beta)
alpha1 <- alpha0 - multiplier*summary(reg0)$coef[1,2]
beta1  <- beta0  - multiplier*summary(reg0)$coef[2,2]*(
              sign(summary(reg0)$cov.unscaled[1,2]))
alpha2 <- alpha0 + multiplier*summary(reg0)$coef[1,2]
beta2  <- beta0  + multiplier*summary(reg0)$coef[2,2]*(
              sign(summary(reg0)$cov.unscaled[1,2]))

# iterate to convergence
conv <- F
while (!conv) {
  # compute weights according to proportion of SSE
  resid1 <- yy - alpha1 - beta1*xx
  resid2 <- yy - alpha2 - beta2*xx
  wt1 <- (resid2**2/(resid1**2 + resid2**2))
  wt2 <- 1 - wt1

  # update estimates
  reg1 <- lm(yy ~ xx, weights=wt1)
  reg2 <- lm(yy ~ xx, weights=wt2)
  alpha1.old <- alpha1
  alpha2.old <- alpha2
  beta1.old  <- beta1
  beta2.old  <- beta2
  alpha1 <- reg1$coefficients[1]
  beta1  <- reg1$coefficients[2]
  alpha2 <- reg2$coefficients[1]
  beta2  <- reg2$coefficients[2]
  conv <- ((abs(alpha1 - alpha1.old) < tolerance) &
           (abs(beta1 - beta1.old)   < tolerance) &
           (abs(alpha2 - alpha2.old) < tolerance) &
           (abs(beta2 - beta2.old)   < tolerance))
             }

  # at convergence we enforce beta1 < beta2 so that component 1 represents
  #   the "responders"
  exchange <- beta1 > beta2
  alpha1.old <- alpha1
  alpha2.old <- alpha2
  beta1.old  <- beta1
  beta2.old  <- beta2
  alpha1 <- ifelse(exchange, alpha2.old, alpha1.old)
  beta1  <- ifelse(exchange, beta2.old, beta1.old)
  alpha2 <- ifelse(exchange, alpha1.old, alpha2.old)
  beta2  <- ifelse(exchange, beta1.old, beta2.old)
  wt1.old <- wt1
  wt2.old <- wt2
  wt1    <- exchange*wt2.old + ((!exchange)*wt1.old)
  wt2    <- exchange*wt1.old + ((!exchange)*wt2.old)

  # proportion of sample weights in Model 1
  propwt1 <- mean(wt1)
  # R-squareds of two- and one-component models
  resid0 <- yy - alpha0 - beta0*xx
  resid1 <- yy - alpha1 - beta1*xx
  resid2 <- yy - alpha2 - beta2*xx
  rsquared.single <- 1 - sum(resid0**2)/sum((yy - mean(yy))**2)
  rsquared.double <- 1 - (sum(wt1*(resid1**2) + wt2*(resid2**2))/
                         sum((yy - mean(yy))**2))
  output <- list(rsquared.single=rsquared.single,
               rsquared.double=rsquared.double,
               alpha0=alpha0, beta0=beta0,
               alpha1=alpha1, beta1=beta1, alpha2=alpha2, beta2=beta2,
               wt1=wt1, wt2=wt2, propwt1=propwt1,
               xx=xx, yy=yy, resid1=resid1, resid2=resid2)
  return(output)                                  }

# Test 1
# Sample size must be even
nn <- 10000
alpha1.test <- 0
beta1.test  <- 1
alpha2.test <- 1
beta2.test  <- -2
scale <- .2

xx <- seq(from=0, to=1, length.out=nn/2)
```

```
yy <- alpha1.test + beta1.test*xx + rnorm(nn/2, 0, 1*scale)
yy <- c(yy, alpha2.test + beta2.test*xx + rnorm(nn/2, 0, 2*scale))
xx <- c(xx, xx)

regtest1 <- mixreg(yy, xx, multiplier=1, tolerance=.000001)

par(mfrow=c(2,1), oma=c(1.5,2,2,1), mar=c(4,4,4,0.5))
# plot data and results
plot(xx, yy, main="Test 1 of two-component mixture regression", pch=".")
abline(regtest1$alpha1, regtest1$beta1, col="red", lwd=2)
abline(regtest1$alpha2, regtest1$beta2, col="red", lwd=2)
abline(alpha1.test, beta1.test, lwd=2, lty=2)
abline(alpha2.test, beta2.test, lwd=2, lty=2)

rr1 <- regtest1
rr1$wt1 <- NULL
rr1$wt2 <- NULL
rr1$xx <- NULL
rr1$yy <- NULL
rr1$resid1 <- NULL
rr1$resid2 <- NULL
print(rr1)

# Test 2
# Sample size must be even
nn <- 10000
alpha1.test <- 0
beta1.test  <- 0
alpha2.test <- 0
beta2.test  <- 0
scale <- 1

xx <- seq(from=0, to=1, length.out=nn/2)
yy <- alpha1.test + beta1.test*xx + rnorm(nn/2, 0, 1*scale)
yy <- c(yy, alpha2.test + beta2.test*xx + rnorm(nn/2, 0, 2*scale))
xx <- c(xx, xx)

regtest2 <- mixreg(yy, xx, multiplier=1, tolerance=.000001)

# plot data and results
plot(xx, yy, main="Test 2 of two-component mixture regression", pch=".")
abline(regtest2$alpha1, regtest2$beta1, col="red", lwd=2)
abline(regtest2$alpha2, regtest2$beta2, col="red", lwd=2)
abline(alpha1.test, beta1.test, lwd=2, lty=2)
abline(alpha2.test, beta2.test, lwd=2, lty=2)

rr2 <- regtest2
rr2$wt1 <- NULL
rr2$wt2 <- NULL
rr2$xx <- NULL
```

```
rr2$yy <- NULL
rr2$resid1 <- NULL
rr2$resid2 <- NULL
print(rr2)

par(mfrow=c(2,1), oma=c(1.5,2,2,1), mar=c(4,4,4,0.5))

# Now fit mixture models to the data

for (i in ii) {
   xx <- as.numeric(evil(i)$FDG0)
   yy <- as.numeric(evil(i)$FDG3)
   uu <- mixreg(yy, xx, multiplier=1, tolerance=.000001)
   assign(paste("mixreg.", i, sep=""), uu)
         }

evil2 <- function(i) eval(parse(text=paste("mixreg.", i, sep="")))
par(mfrow=c(2,1), oma=c(1.5,2,2,1), mar=c(4,4,4,0.5))

for (i in (1:length(ii))) {
   xx <- as.numeric(evil(ii[i])$FDG0)
   yy <- as.numeric(evil(ii[i])$FDG3)
   uu <- range(xx, yy)
   plot(xx, yy, xlim=uu, ylim=uu, pch=".",
      main=paste("Two-component mixture reg. for CDP00", jj[i], sep=""),
      xlab="FDG0", ylab="FDG3")
   mtext("FDG3 vs. FDG0", outer=T, cex=2, col="red")
   mm <- evil2(ii[i])
   abline(mm$alpha1, mm$beta1, col="red", lwd=2)
   abline(mm$alpha2, mm$beta2, col="red", lwd=2)
   abline(mm$alpha0, mm$beta0, lwd=2, lty=2)
                  }

rsquared.single.all <- NULL
rsquared.double.all <- NULL
alpha0.all <- NULL
beta0.all <- NULL
alpha1.all <- NULL
beta1.all <- NULL
alpha2.all <- NULL
beta2.all <- NULL
propwt1.all <- NULL

for (i in (1:length(ii))) {
rr <- evil2(ii[i])
rsquared.single.all <- c(rsquared.single.all, rr$rsquared.single)
rsquared.double.all <- c(rsquared.double.all, rr$rsquared.double)
alpha0.all <- c(alpha0.all, rr$alpha0)
beta0.all <- c(beta0.all, rr$beta0)
alpha1.all <- c(alpha1.all, rr$alpha1)
beta1.all <- c(beta1.all, rr$beta1)
alpha2.all <- c(alpha2.all, rr$alpha2)
beta2.all <- c(beta2.all, rr$beta2)
propwt1.all <- c(propwt1.all, rr$propwt1)
                        }

uu <- cbind(rsquared.single.all, rsquared.double.all,
           alpha0.all, beta0.all, alpha1.all, beta1.all, alpha2.all, beta2.all,
           propwt1.all)
dimnames(uu)[[1]] <- jj
round(uu,3)
```